Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 10 of 137 returned.

☐ 1. Document ID: US 6924807 B2

L5: Entry 1 of 137 File: USPT Aug 2, 2005

US-PAT-NO: 6924807
DOCUMENT-IDENTIFIER: US 6924807 B2

TITLE: Image processing apparatus and method

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Image

☐ 2. Document ID: US 6919900 B2

L5: Entry 2 of 137 File: USPT Jul 19, 2005

US-PAT-NO: 6919900
DOCUMENT-IDENTIFIER: US 6919900 B2

TITLE: Methods and systems for preparing graphics for display on a computing device

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Image

☐ 3. Document ID: US 6919892 B1

L5: Entry 3 of 137 File: USPT Jul 19, 2005

US-PAT-NO: 6919892
DOCUMENT-IDENTIFIER: US 6919892 B1

TITLE: Photo realistic talking head creation system and method

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Image

☐ 4. Document ID: US 6870538 B2

L5: Entry 4 of 137 File: USPT Mar 22, 2005

US-PAT-NO: 6870538
DOCUMENT-IDENTIFIER: US 6870538 B2

TITLE: Video and graphics system with parallel processing of graphics windows

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Image

☐ 5. Document ID: US 6854003 B2

L5: Entry 5 of 137 File: USPT Feb 8, 2005

US-PAT-NO: 6854003
DOCUMENT-IDENTIFIER: US 6854003 B2

TITLE: Video frame rendering engine

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Image

---

☐ 6. Document ID: US 6853385 B1

L5: Entry 6 of 137 File: USPT Feb 8, 2005

US-PAT-NO: 6853385
DOCUMENT-IDENTIFIER: US 6853385 B1

TITLE: Video, audio and graphics decode, composite and display system

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Image

---

☐ 7. Document ID: US 6813777 B1

L5: Entry 7 of 137 File: USPT Nov 2, 2004

US-PAT-NO: 6813777
DOCUMENT-IDENTIFIER: US 6813777 B1

TITLE: Transaction dispatcher for a passenger entertainment system, method and article of manufacture

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Image

---

☐ 8. Document ID: US 6798420 B1

L5: Entry 8 of 137 File: USPT Sep 28, 2004

US-PAT-NO: 6798420
DOCUMENT-IDENTIFIER: US 6798420 B1

TITLE: Video and graphics system with a single-port RAM

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Image

---

☐ 9. Document ID: US 6791559 B2

L5: Entry 9 of 137 File: USPT Sep 14, 2004

US-PAT-NO: 6791559
DOCUMENT-IDENTIFIER: US 6791559 B2

TITLE: Parameter circular buffers

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw Desc | Image

10. Document ID: US 6771264 B1

L5: Entry 10 of 137 File: USPT Aug 3, 2004

US-PAT-NO: 6771264
DOCUMENT-IDENTIFIER: US 6771264 B1
** See image for Certificate of Correction **

TITLE: Method and apparatus for performing tangent space lighting and bump mapping in a deferred shading graphics processor

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw Desc | Image

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Terms	Documents
L4 and clock	137

Display Format: TI Change Format

Previous Page Next Page Go to Doc#

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 10 of 11 returned.

☐ 1. Document ID: US 6854003 B2

L6: Entry 1 of 11 File: USPT Feb 8, 2005

US-PAT-NO: 6854003
DOCUMENT-IDENTIFIER: US 6854003 B2

TITLE: Video frame rendering engine

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Image

☐ 2. Document ID: US 6593929 B2

L6: Entry 2 of 11 File: USPT Jul 15, 2003

US-PAT-NO: 6593929
DOCUMENT-IDENTIFIER: US 6593929 B2

TITLE: High performance low cost video game system with coprocessor providing high speed efficient 3D graphics and digital audio signal processing

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Image

☐ 3. Document ID: US 6556197 B1

L6: Entry 3 of 11 File: USPT Apr 29, 2003

US-PAT-NO: 6556197
DOCUMENT-IDENTIFIER: US 6556197 B1

TITLE: High performance low cost video game system with coprocessor providing high speed efficient 3D graphics and digital audio signal processing

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Image

☐ 4. Document ID: US 6342892 B1

L6: Entry 4 of 11 File: USPT Jan 29, 2002

US-PAT-NO: 6342892
DOCUMENT-IDENTIFIER: US 6342892 B1

TITLE: Video game system and coprocessor for video game system

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Image

☐ 5. Document ID: US 6331856 B1

L6: Entry 5 of 11 File: USPT Dec 18, 2001

US-PAT-NO: 6331856
DOCUMENT-IDENTIFIER: US 6331856 B1

TITLE: Video game system with coprocessor providing high speed efficient 3D graphics and digital audio signal processing

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw Desc | Image

☐ 6. Document ID: US 6239810 B1

L6: Entry 6 of 11 File: USPT May 29, 2001

US-PAT-NO: 6239810
DOCUMENT-IDENTIFIER: US 6239810 B1

TITLE: High performance low cost video game system with coprocessor providing high speed efficient 3D graphics and digital audio signal processing

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw Desc | Image

☐ 7. Document ID: US 6166748 A

L6: Entry 7 of 11 File: USPT Dec 26, 2000

US-PAT-NO: 6166748
DOCUMENT-IDENTIFIER: US 6166748 A

TITLE: Interface for a high performance low cost video game system with coprocessor providing high speed efficient 3D graphics and digital audio signal processing

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw Desc | Image

☐ 8. Document ID: US 6064393 A

L6: Entry 8 of 11 File: USPT May 16, 2000

US-PAT-NO: 6064393
DOCUMENT-IDENTIFIER: US 6064393 A
** See image for Certificate of Correction **

TITLE: Method for measuring the fidelity of warped image layer approximations in a real-time graphics rendering pipeline

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw Desc | Image

☐ 9. Document ID: US 6016150 A

L6: Entry 9 of 11 File: USPT Jan 18, 2000

US-PAT-NO: 6016150
DOCUMENT-IDENTIFIER: US 6016150 A
** See image for Certificate of Correction **

TITLE: Sprite compositor and method for performing lighting and shading operations using a compositor to combine factored image layers

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Image

10. Document ID: US 5852449 A

L6: Entry 10 of 11 File: USPT Dec 22, 1998

US-PAT-NO: 5852449
DOCUMENT-IDENTIFIER: US 5852449 A

TITLE: Apparatus for and method of displaying running of modeled system designs

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Image

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Terms	Documents
L5 and animation.ab.	11

Display Format: TI Change Format

Previous Page Next Page Go to Doc#

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 11 through 11 of 11 returned.

☐ 11. Document ID: US 4710877 A

L6: Entry 11 of 11 File: USPT Dec 1, 1987

US-PAT-NO: 4710877
DOCUMENT-IDENTIFIER: US 4710877 A
** See image for Certificate of Correction **

TITLE: Device for the programmed teaching of arabic language and recitations

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Image

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Terms	Documents
L5 and animation.ab.	11

Display Format: TI Change Format

Previous Page Next Page Go to Doc#

Freeform Search

Database: US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term: L5 and animation.ab.

Display: 10 **Documents in Display Format:** TI **Starting with Number** 1

Generate: ○ Hit List ◉ Hit Count ○ Side by Side ○ Image

Search Clear Interrupt

Search History

DATE: Saturday, October 01, 2005 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L6	L5 and animation.ab.	11	L6
L5	L4 and clock	137	L5
L4	L3 and synchronization	225	L4
L3	L2 and interactive	421	L3
L2	L1 and refresh	843	L2
L1	animation	9856	L1

END OF SEARCH HISTORY

L5: Entry 1 of 137 File: USPT Aug 2, 2005

DOCUMENT-IDENTIFIER: US 6924807 B2
TITLE: Image processing apparatus and method

Brief Summary Text (15):
In accordance with at least one further aspect of the present invention, an apparatus for processing image data to produce an image for display on a display screen, includes a plurality of graphics processors, each graphics processor being operable to render the image data into frame image data and to store the frame image data in a respective local frame buffer, and each graphics processor including a synchronization counter operable to produce a local synchronization count indicating when the frame image data should be released from the local frame buffer; a control processor operable to provide instructions to the plurality of graphics processors; and at least one merge unit operable to synchronously receive the frame image data from the respective local frame buffers and to synchronously produce combined frame image data based thereon.

Brief Summary Text (17):
Preferably, the core merge unit transmits the merge synchronization signal to the respective graphics processors by way of the bus controller and the data bus. Further, it is preferred that the control processor communicates at least the instructions concerning the one or more modes of operation to the respective local merge units unit by way of the bus controller and the control data bus.

Brief Summary Text (20):
Preferably, the control processor communicates at least the instructions concerning the one or more modes of operation to the respective local merge units and the core merge unit by way of packet switch, the packet switch controller and the control data bus. Further, it is preferred that the core merge unit transmits the merge synchronization signal to the respective graphics processors by way of the packet switch controller, and the packet switch.

Brief Summary Text (24):
In accordance with at least one further aspect of the present invention, a method for processing image data to produce an image for display on a display screen, includes: rendering the image data into frame image data using a plurality of graphics processors; storing the frame image data in respective local frame buffers; producing respective local synchronization counts indicating when the frame image data should be released from the respective local frame buffers; and synchronously producing combined frame image data from the frame image data.

Detailed Description Text (3):
The apparatus 100 preferably includes a control processor 102 (and associated memory), a plurality of graphics processors 104, at least one merge unit 106, and a synchronization unit 108. The control processor 102 preferably communicates with the plurality of graphics processors 104, the merge unit 106, and the synchronization unit 108 by way of bus 126. In accordance with at least one aspect of the present invention, separate data lines 127 couple each of the graphics processors 104 to the merge unit 106 such that exclusive communication between the respective graphics processors 104 and the merge unit 106 is obtained. Further, separate synchronization lines 129 preferably couple the synchronization unit 108 to each of the graphics processors 104 and the merge unit 106 such that exclusive communication therebetween is obtained.

Detailed Description Text (4):
At least some, and preferably each, of the graphics processors 104 is operable to render image data into frame image data (e.g., pixel data) and to store the frame image data in a respective local frame buffer 112. More particularly, at least some of the graphics processors 104

preferably include a rendering unit 110 for performing the rendering function and a local frame buffer 112 for at least temporarily storing the frame image data. In addition, it is preferred that each graphics processor 104 include a processing unit 114 operable to facilitate performing various processes on the image data, such as polygonal spatial transformation (e.g., translation, rotation, scaling, etc.); 3D to 2D transformation (e.g., perspective view transformation, etc.); and generating sequence instructions (e.g., polygon instructions). The graphics processors 104 preferably also each include an input/output interface 116 suitable for facilitating communication over the bus 126. The graphics processors 104 may each include an optional local synchronization circuit 118, which will be discussed in more detail hereinbelow.

Detailed Description Text (6):
The merge unit 106 is preferably operable to synchronously receive frame image data from each of the respective local frame buffers 112 and to synchronously produce combined frame image data based thereon. Preferably, the frame image data are synchronously received by the merge unit 106 in response to a merge synchronization signal produced by one of the merge unit 106 and the synchronization unit 108 and received by the graphics processors 104 over synchronization lines 129. Indeed, the plurality of graphics processors 104 preferably synchronously release the frame image data from the respective local frame buffers 112 to the merge unit 106 in response to the merge synchronization signal.

Detailed Description Text (7):
Preferably, the merge synchronization signal is synchronized in accordance with a display protocol (or standard) defining how respective frames of the combined frame image data are to be displayed. These display protocols may include the well known NTSC protocol, the HDTV protocol, the 35 mm movie protocol, the PAL protocol, etc. The display protocol defines, among other things, the frame rate at which successive frames of the combined image data are to be displayed, and a blanking period, defining when the frames of the combined frame image data are to be refreshed. For example, the blanking period may dictate how long a given frame of the combined frame image data should dwell on the display prior to refresh (as is the case with the NTSC protocol). Alternatively, the blanking period may dictate when a given frame of the combined frame image data should be removed from the display prior to refresh (as is the case with the 35 mm movie protocol).

Detailed Description Text (10):
Advantageously, producing the combined frame image data from the merge unit 106 in synchronous fashion with the blanking period of desired display protocols permits the integration of frame image data from different display protocols and/or from different sources of the frame image data. For example, frame image data from the 35 mm film display protocol may be combined with frame image data produced by way of computer graphics. In a broad sense, therefore, the merge synchronization signal is either derived from or sets a desired frame rate at which the combined frame image data are provided. When the combined frame image data are to be displayed on a CRT display screen, the merge synchronization signal is preferably synchronized with the vertical blanking period (or frame rate) of a CRT display screen. Alternatively, when the combined frame image data are to be merged with or utilized to produce a moving image consistent with a 35 mm film cinematography, the merge synchronization signal is preferably synchronized with the frame rate (or blanking period) consistent with that protocol.

Detailed Description Text (11):
Reference is now made to FIG. 4, which is a timing diagram that illustrates the relationship between the operation of the apparatus 100 of FIG. 3 and certain aspects of a display protocol. For example the relationship between the blanking period, the merge synchronization signal, the rendering period, and the merge period is shown. As illustrated at the top timing waveform, a frame includes an interval during which an image frame is to be displayed or refreshed (indicated by a low logic state) and a period during which an image frame is to be made ready for display, e.g., the blanking period (indicated by a high logic state).

Detailed Description Text (12):
In accordance with at least one aspect of the present invention, the merge synchronization signal 132 includes transitions, such as one of rising and falling edges, that are proximate to the ends 142, 144 of the blanking periods (i.e., the transitions from high logic levels to low logic levels). For example, the merge synchronization signal 132 may include transitions that lead the ends 142, 144 of the blanking periods. Alternatively, the merge synchronization signal 132 may include transitions that are substantially coincident with the ends 142, 144 of the

blanking periods. In any case, the merge synchronization signal 132 is preferably synchronized with the frame rate dictated by the display protocol such that the merge unit 106 initiates the production of, or releases, the combined frame image data for display at the end of at least one of the blanking periods.

Detailed Description Text (13):
In order for the local frame buffers 112 to be ready to release the combined frame image data for display at the ends 142, 144 of the blanking periods, at least some of the graphics processors 104 preferably initiate rendering the image data into the frame buffers 112 prior to the ends 142, 144 of the blanking periods, such as at 136A-D. As shown, four rendering processes are initiated at respective times 136A-D, although any number of rendering processes may be employed using any number of graphics processors 104 without departing from the scope of the invention. Further, the rendering processes in each of the graphics processors 104 may be initiated at the same time or may be initiated at different times. In accordance with at least one aspect of the present invention, the rendering units 110 of the graphics processors 104 are preferably operable to begin rendering the image data into the respective frame buffers 112 asynchronously with respect to the merge synchronization signal 132. For example, the control processor 102 may issue a "begin rendering" command (or trigger) to the graphics processors 104, where the begin rendering command is asynchronous with respect to the merge synchronization signal 132. The begin rendering command may be given any suitable name, such as DRAWNEXT. It is also contemplated that the processing units 114 may initiate rendering as a result of software program execution without specific triggering from the control processor 102. Each (or a group) of the graphics processors 104 may complete the rendering process of the image data into the respective frame buffers 112 at any time (e.g., at 138A-D). It is preferred that at least one of the graphics processors 104 is operable to issue a rendering complete signal to the control processor 102 when it has completed rendering a frame of the frame image data. The rendering complete command may be given any suitable name, such as DRAWDONE.

Detailed Description Text (14):
In response to the merge synchronization signal 132, the respective graphics processors 104 preferably release the frame image data from the respective local frame buffers 112 to the merge unit 106 such that the merge unit 106 may synchronously produce the combined frame image data therefrom. The combined frame image data are preferably produced by the merge unit 106 during the interval between the end of one blanking period and the beginning of the next blanking period as shown by the logic transitions labeled 140.

Detailed Description Text (15):
With reference to FIGS. 3 and 4, one or more of the graphics processors 104 preferably include a respective local synchronization circuit 118 operable to receive the merge synchronization signal 132 over synchronization lines 129 from the synchronization unit 108 or from the merge unit 106. The respective local synchronization units 118 are preferably further operable to deliver the synchronization signal to the various portions of the graphics processors 104 to enable them to synchronously release the respective frame image data from the local frame buffers 112 to the merge unit 106. The local synchronization circuits 118 preferably employ counters to count between respective transitions of the merge synchronization signal 132.

Detailed Description Text (17):
Another of these modes preferably provides that one of more of the graphics processors 104 completes rendering the image data into the respective frame buffers 112 prior to the end of an integral number of blanking periods. This mode is illustrated in FIG. 5, which is a timing diagram showing the relationship between the frame rate, the blanking period, the merge synchronization signal 132, the rendering period, and the merge period carried out by the apparatus 100 of FIG. 3. The plurality of graphics processors 104 may initiate the rendering of the image data into the respective local frame buffers 112 (e.g., at 136) either synchronously, asynchronously, in response to a begin rendering command (DRAWNEXT) from the control processor 102 or processing units 114, etc., it being most preferred that at least a group of the graphics processors 104 initiate rendering at substantially the same time and at some point during a blanking period. Each of the graphics processors 104 may complete the rendering process at different times, such as at 138A-D.

Detailed Description Text (18):
According to this mode of operation, however, the rendering processes need not be completed prior to the termination of each blanking period; rather, they may be completed prior to the termination of an integral number of blanking periods (such as two blanking periods as shown).

Although at least some of the graphics processors 104 may not complete a given rendering process prior to the end 142 of a first blanking period, they may complete the rendering process prior to the end 144 of a second blanking period. Advantageously, in this mode of operation, each of the graphics processors 104 enjoys the benefit of additional time during which to render the image data into the respective local frame buffers 112. In response to the merge synchronization signal 132, the respective graphics processors 104 preferably release the frame image data from the respective local frame buffers 112 to the merge unit 106 at the end of every second blanking period (such as at 144). New image data may be rendered into the local frame buffers 112 beginning at, for example, 146. Although in the example illustrated in FIG. 5, the integral number of blanking periods is two, any integral number may be employed in this mode of operation without departing from the scope of the invention.

Detailed Description Text (20):
Similarly, the second graphics processor 104 should complete rendering the image data into local frame buffer 112B prior to the end 142 of the first blanking period (e.g., at 138B), while the second graphics processor 104 need not complete rendering the image data into local frame buffer 112B' until prior to the end 144 of the second blanking period (e.g., at 138B'). As frame image data are available for release from the local frame buffer 112A and the local frame buffer 112B prior to the end 142 of the first blanking period, such frame image data are preferably released in response to the merge synchronization signal 132 to the merge unit 106 as shown in FIG. 6A at 140AB. As frame image data are available in each of local frame buffer 112A' and local frame buffer 112B' prior to the end 144 of the second blanking period, such frame image data are preferably released in response to the merge synchronization signal 132 to the merge unit 106 as shown at 140A'B'.

Detailed Description Text (23):
With reference to FIGS. 7A-E, the modes of operation in accordance with one or more aspects of the present invention preferably include at least one of area division, averaging, layer blending, Z-sorting and layer blending, and flip animation. With reference to FIG. 7A, the area division mode of operation preferably provides that at least two of the local frame buffers 112 (as shown, four local frame buffers 112A-D are contemplated) are partitioned into respective rendering areas 120A-D that correspond with respective areas of the display screen that will be covered by the combined frame image data 122, and non-rendering areas 124A-D that are not utilized to carry frame image data. In accordance with the area division mode, an aggregate of the rendering areas 120A-D results in a total rendering area that corresponds with a total area of the display screen that will be covered by the combined frame image data 122. In this mode, the merge unit 106 is preferably operable to synchronously aggregate the respective frame image data from the respective rendering areas 120A-C of the graphics processors 104 based on the known alpha blending technique to produce the combined frame image data. The local frame buffers 112A-D may be employed by separate graphics processors 104 or may be employed by, and distributed among, a lesser number of graphics processors 104. The area division mode of operation may take advantage of one or more of the timing relationships discussed hereinabove with respect to FIGS. 4, 5 and 6A, it being preferred that the one or more graphics processors 104 complete rendering the image data into the respective rendering areas 120A-D of the local frame buffers 112A-D prior to the end of each blanking period (for example, see FIG. 4).

Detailed Description Text (27):
With reference to FIG. 7E, the flip animation mode preferably provides that the local frame buffers 112 (as shown, four local frame buffers 112A-D are contemplated) of at least two graphics processors 104 include frame image data that are capable of covering the total area covered by the combined frame image data and that the merge unit 106 is operable to produce the combined frame image data by sequentially releasing the respective frame image data from the local frame buffers 112 of the graphics processors 104. The flip animation mode may employ one or more of the timing relationships discussed hereinabove with respect to FIGS. 4, 5, and 6A, it being preferred that the graphics processors 104 complete rendering the image data into the respective frame buffers 112A-D prior to the ends of an integral number of blanking periods, where the integral number of blanking periods corresponds to the number of graphics processors 104 participating in the flip animation mode (for example, see FIGS. 5 and 6A). Alternatively, the integral number of blanking periods may correspond to the number of local frame buffers 112 participating in the flip animation mode.

Detailed Description Text (31):
With reference to FIGS. 3, 4, and 9, the high level flow diagram of FIG. 9 provides additional details concerning one or more further aspects of the apparatus 110 of the present invention.

Actions 170 and 172 relate to initializing the plurality of graphics processors 104, the merge unit 106, and the synchronization unit 108. In particular, at action 170, the control processor 102 preferably transmits program data and format data to the graphics processors 104. In accordance with the invention, the program data and the format data preferably imbue the graphics processors 104 with the necessary capability to carry out one or more of the modes of operation discussed hereinabove with respect to FIGS. 4-7.

Detailed Description Text (33):
The format data preferably relate to at least one of the modes of operation (e.g., the timing relationships, area division, averaging, layer blending, z-sorting and layer blending, flip animation, etc.); display protocol information, such as the NTSC protocol, the HDTV protocol, the 35 mm film cinematography protocol, etc. More particularly, the display protocol may include information concerning the desired frame rate, blanking period, the display size, the display aspect ratio (e.g., the ratio between the display height to width), the image resolution, etc. In general, the format data ensure that the graphics processors 104 are configured to execute a particular mode or modes of operation before called upon to render the image data or release frame image data to the merge unit 106. Preferably, the format data are transmitted to the plurality of graphics processors 104 by way of the bus 126.

Detailed Description Text (35):
The graphics processors 104 preferably transmit system ready signals to at least one of the respective local synchronization units 118 and the control processor 102 when they have initialized themselves consistent with the program data and format data (action 174). The system ready signals may have any suitable name, such as SYSREADY. The system ready signals are preferably transmitted to the control processor 102 by way of the bus 126.

Detailed Description Text (36):
The synchronization circuit 108 preferably periodically transmits the merge synchronization signal 132 to the control processor 102 and to the graphics processors 104. Alternatively, the merge unit 106 periodically transmits the merge synchronization signal 132 to the graphics processors 104. In either case, the merge synchronization signal 132 is synchronized to a desired frame rate and blanking period associated with the desired display protocol (action 176). The merge synchronization signal 132 is preferably transmitted to the graphic processors 104 by way of the dedicated synchronization lines 129 (or by way of the merge unit 106) and to the control processor 102 by way of the bus 126. When the graphics processors 104 include a local synchronization circuit 118, they preferably count a predetermined number of clock signals (or cycles) between transitions of the merge synchronization signal 132 to ensure that the frame image data are released at the proper time (action 178).

Detailed Description Text (37):
At action 180, the control processor 102 preferably transmits a begin rendering command instruction (or trigger) DRAWNEXT indicating that the graphics processors 104 should initiate the rendering of the image data into the respective local frame buffers 112. This trigger may be transmitted to the graphics processors 104 over the bus 126, it being preferred that the trigger is issued through the synchronization unit 108 to the respective local synchronization units 118 of the graphics processors 104. As discussed above, however, the graphics processors 104 need not receive an explicit trigger from the control processor 102 if an application software program running locally on the processing units 114 provides a begin rendering command itself. Certain advantages, however, are achieved when the control processor 102 issues a trigger to at least some of the graphics processors 104 indicating when the graphics processors 104 should begin rendering the image data into the respective local frame buffers 112. At action 182, the respective graphics processors 104 perform the rendering function on the image data to produce the frame image data and store the same in the respective local frame buffers 112.

Detailed Description Text (39):
At action 186, at least one of the graphics processors 104 preferably issues a signal to the control processor 102 indicating that rendering is complete (e.g., DRAWDONE). The rendering complete signal may be transmitted to the control processor 102 by way of the bus 126, it being preferred that the signal is transmitted through the local synchronization circuit 118 to the synchronization unit 108 prior to being delivered to the control processor 102.

Detailed Description Text (40):
At action 188, the appropriate graphics processors 104 release the frame image data from the

respective local frame buffers 112 to the merge unit 106 (e.g., when the merge synchronization signal 132 indicates that the end of an appropriate blanking period has been reached) and the merge unit 106 produces the combined frame image data for display (action 190). At least some of these processes are repeated frame-by-frame to produce a high quality moving image for display on the display screen.

Detailed Description Text (41):
Advantageously, the apparatus 100 readily lends itself to scalability when additional data throughput is required. Referring now to FIG. 10, an apparatus 200 for processing image data to produce an image for display in accordance with one or more further aspects of the present invention is shown. The apparatus 200 includes a control processor 102, a plurality of graphics apparatuses 100A-D, a core merge unit 106N, and a core synchronization unit 108N. The respective graphics apparatuses 100A, 100B, 100C, etc. in FIG. 10 are preferably substantially similar to the apparatus 100 shown in FIG. 3. The merge unit 106 of FIG. 3 corresponds to respective local merge units 106A, 106B, 106C, etc. that are coupled to respective sets of graphics processors 104. The local merge units 106A, 106B, 106C, etc. each preferably produce local combined frame image data consistent with the modes of operation discussed hereinabove. Each of the local merge units 106A, 106B, 106C, etc. are preferably coupled to the core merge unit 106N, which is operable to synchronously receive the local combined frame image data and to synchronously produce combined frame image data to be displayed.

Detailed Description Text (42):
The core merge unit 106N is preferably operable to produce the merge synchronization signal 132 and each apparatus 100A, 100B, 100C, etc. preferably includes a local synchronization unit 108A, 108B, 108C, etc., respectively, (that corresponds to the synchronization unit 108 of FIG. 3). The respective local synchronization units 108A, 108B, 108C, etc. preferably utilize the merge synchronization signal 132 to ensure that the frame image data are synchronously released to each of the local merge units 106A, 106B, 106C, etc. and ultimately released to the core merge unit 106N.

Detailed Description Text (43):
Preferably, the respective local frame buffers 112 of the respective groups of graphics processors 104A-D are operatively coupled to the respective local merge units 106A-D by way of separate data lines 127 such that exclusive communication between the respective local frame buffers and the respective local merge units is obtained (e.g., such that exclusive transmission of the frame image data to the respective local merge units 106A, 106B, 106C, etc. is obtained). Preferably, separate data lines 127A-D couple the respective local merge units 106A-D to the core merge unit 106N such that exclusive communication therebetween is obtained (e.g., such that exclusive transmission of the local combined frame image data from the respective local merge units 106A-D to the core merge unit 106N is obtained). Further, separate synchronization lines preferably couple the respective local synchronization units 108A-D to the core synchronization unit 108N such that exclusive communication therebetween is obtained. Although not explicitly shown, a bus, substantially similar to the bus 126 of FIG. 3, preferably extends from the control processor 102 to each of the graphics processors 104A, 104B, 104C, etc., the core merge unit 106N, and the core synchronization unit 108N.

Detailed Description Text (46):
Reference is now made FIG. 12, which illustrates an apparatus 300 for processing image data to produce an image for display in accordance with one or more further aspects of the present invention. The apparatus 300 preferably includes a control processor 302, a plurality of graphics processors 304, and a merge unit 306. The control processor 302 is preferably operable to provide instructions to the plurality of graphics processors 304. The control processor 302 preferably includes a master timing generator 308, a controller 310, and a counter 312. The master timing generator 308 is preferably operable to produce a synchronization signal that is synchronized with blanking periods of a display protocol, such as one or more of the display protocols discussed hereinabove. The controller 310 is preferably operable to provide instructions to the plurality of graphics processors 304 similar to the instructions that the control processor 102 of FIG. 3 provides to the graphics processors 104 of the apparatus 100. In addition, the control processor 302 preferably provides reset signals to the plurality of graphics processors 304 (utilized in combination with the synchronization signal) to synchronize the release of frame image data to the merge unit 306 as will be discussed in more detail below.

Detailed Description Text (47):

Each of the graphics processors 304 preferably includes a rendering unit 314 operable to render image data into frame image data and to store the frame image data in a respective local frame buffer (not shown). Indeed, each graphics processor 304 preferably includes at least some of the functional blocks employed in the graphics processors 104 of FIG. 3. In addition, each graphics processor 304 preferably includes a synchronization counter 318 operable to produce a local synchronization count indicating when the frame image data should be released from the local frame buffer. The respective synchronization counters 318 preferably one of increment and decrement their respective synchronization counts based on the synchronization signal from the timing generator 308. The graphics processors 304 are preferably operable to release their respective frame image data to the merge unit 306 when their respective local synchronization counts reach a threshold. The reset signals issued by the control 310 of the control processor 302 are preferably utilized to reset the synchronization counters 318 of the respective graphics processors 304. In this way, the control processor 302 is operable to manipulate the timing of the release of the frame image data from the respective rendering units 314. Thus, the apparatus 300 may facilitate the modes of operation discussed hereinabove with respect to FIGS. 4-7, e.g., the timing relationships, area division, averaging, layer blending, z-sorting and layer blending, and flip animation.

Detailed Description Text (48):
With reference to FIG. 13A, timing relationships between the synchronization signal issued by the timing generator 308, the reset signal, and a respective one of the synchronization counters 318 are illustrated. In particular, the synchronization counter 318 increments (or decrements) on transitions of the synchronization signal. The reset signal, however, resets the synchronization counter 318 to begin counting from a predetermined level, such as zero. With reference to FIG. 13B, an alternative configuration may be employed to achieve a finer timing accuracy. In particular, the synchronization counters 318 may be substituted by, or operate in conjunction with, sub-synchronization counters (not shown) and the timing generator 308 may produce a sub-synchronization signal operating at a higher frequency than the synchronization signal. The sub-synchronization counters preferably one of increment and decrement in response to transitions of the sub-synchronization signal and are reset upon receipt of the reset signal from the control processor 302. Using this arrangement, the graphics processors 304 are preferably operable to release their respective frame image data to the merge unit 306 when the respective sub-synchronization counters reach a threshold (which would be higher than the threshold employed using the timing of FIG. 13A).

Detailed Description Text (57):
Reference is now made to FIG. 16, which is a block diagram illustrating an apparatus 400 suitable for use in accordance with one or more further aspects of the present invention. In many respects, the apparatus 400 is substantially similar to the apparatus 200 of FIG. 10. In particular, the apparatus 400 includes a control processor 102, a plurality of graphics apparatuses 100A-D, and a core merge unit 106N. Notably, the apparatus 400 does not require a core synchronization unit 108N as did the apparatus 200 of FIG. 10. The functionality of these functional and/or circuit blocks are substantially similar to those discussed hereinabove with respect to FIGS. 3 and 10 and, therefore, will not be repeated here.

Detailed Description Text (60):
As discussed above with respect to apparatus 200 of FIG. 10 and apparatus 100 of FIG. 3, the core merge unit 106N is preferably operable to produce the merge synchronization signal 132 to facilitate synchronous release of the frame image data, the release of the local combined frame image data, and the production of the combined frame image data. The core merge unit 106N may transmit the merge synchronization signal 132 to the respective graphics processors 104 by way of the control data bus 404, and the data bus 126. In order to ensure that the merge synchronization signal 132 is transmitted quickly and received promptly by the apparatuses 100A-D, the bus controller function 402 is preferably operable to seize control of the data bus 126 upon receipt of the merge synchronization signal 132 over the control data bus 404 and transmit the merge synchronization signal 132 to the respective apparatuses 100A-D on a priority basis. Thus, the bus controller function 402 provides the function of a master bus controller. Alternately, the core merge unit 106N may transmit the merge synchronization signal 132 to the respective graphics processors 104 by way of the control data bus 404 and the local merge units 106A-D.

Detailed Description Text (66):
The control processor 102 is preferably operable to instruct the graphics processors 104, the local merge units 106A-D, and the core merge unit 106N to operate in the one or more modes of

operation on a frame-by-frame basis as discussed in detail hereinabove with respect to other configurations of the invention. In particular, the control processor 102 preferably communicates instructions concerning the one or more modes of operation to the respective destination function and/or circuit blocks by way of the packet switch 504, the instructions having been packetized in accordance with the TCP layer. Further, the core merge unit 106N is preferably operable to produce the merge synchronization signal 132 as discussed in detail hereinabove. The core merge unit 106N may be operable to transmit the merge synchronization signal 132 to various destination functional and/or circuit blocks by way of the control data bus 404, the packet switch controller function 502, and the packet switch 504.

Detailed Description Text (67):
The packet switch controller function 504 may be capable of seizing control of the packet switch 504 when the core merge unit 106N transmits the merge synchronization signal 132 over the control data bus 404 such that the merge synchronization signal 132 may be promptly transmitted to the destination functional and/or circuit blocks. When the TCP layer is capable of transmitting very quickly and guarantees the routing of all packets of an instruction to any of the destination functional and/or circuit blocks within a given latency, then the packet switch controller 502 need not seize the packet switch 504. Rather, a design assumption may be made that the merge synchronization signal 132 will be received by the destination functional and/or circuit blocks (such as the graphics processors 104) within an acceptable tolerance. Alternately, the core merge unit 106N could transmit the merge synchronization signal 132 to the respective graphics processors 104 by way of the control data bus 404 and the local merge units 106A-D.

Detailed Description Text (68):
Reference is now made to FIG. 18, which is block diagram of an apparatus 600 for processing image data to produce an image on a display in accordance with one or more further aspects of the present invention. The apparatus 600 preferably includes a plurality of processing nodes 602, at least one of which is a control node 604, that are coupled together on a packet-switched network, such as a local area network (LAN). The packet-switched network (here a LAN) and an associated TCP layer is assumed to enjoy a data transmission rate (from source node to destination node) that is sufficiently high to support the synchronization schema discussed hereinabove to produce the image.

Detailed Description Text (75):
At action 664, the control node is preferably operable to determine which of the processing nodes 602 are to be retained in the subset of processing nodes to participate in processing the image data. This determination is preferably based on at least one of the data processing capabilities, format information, availability, etc. provided by the processing nodes 602 in response to the node configuration requests. At action 660, the apparatus 600 preferably operates in substantial accordance with the process flow discussed hereinabove with respect to FIG. 9, particularly in terms of the use of the merge synchronization signal to synchronize the release of frame image data and combined frame image data to the one or more merge units 106 of the at least one merge node 608, it being understood that the flow of data between the graphics processors 104, merge units 106, control processor 102, synchronization units 108, etc. are facilitated over the packet-switched network. Moreover, the apparatus 600 of FIG. 18 is preferably operable to provide the requisite resources to implement any of the previously discussed apparatuses hereinabove, for example, apparatus 100 of FIG. 3, apparatus 200 of FIG. 10, apparatus 350 of FIG. 14, apparatus 360 of FIG. 15, apparatus 400 of FIG. 16, apparatus 500 of FIG. 17, or any combination thereof.

Detailed Description Text (78):
It is preferred that the processing nodes of apparatus 700 are coupled together on an open packet-switched network, such as the Internet. The packet-switched network, preferably employs suitable hardware and a TCP layer that enjoys a data transmission rate (e.g., from source node to destination node) that is sufficiently high to support the synchronization schema discussed hereinabove to produce the image.

Other Reference Publication (2):
S. Molnar, J. Eyles, and J. Poulton. PixelFlow: High-Speed Rendering Using Image Composition. In Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, vol. 26 issue 2, pp. 231-240, Jul. 1992.

Other Reference Publication (3):

S. Nishimura and T. Kunii. VC-1: A Scalable Graphics Computer with Virtual Local Frame buffers. In Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, pp. 365-373, Aug. 1996.

CLAIMS:

1. An apparatus for processing image data to produce an image for covering an image area, comprising: a plurality of graphics processors, each graphics processor being operable to render the image data into frame image data and to store the frame image data in a respective local frame buffer; a control processor operable to provide instructions to the plurality of graphics processors; one or more merge units operable to synchronously receive the frame image data from the respective local frame buffers and to synchronously produce combined frame image data based thereon, wherein at least one of the one or more merge units is operable to produce a merge synchronization signal used by the graphics processors to release the frame image data from the respective local frame buffers to the one or more merge units; means for transmitting the entire combined frame image data to a single display; wherein the single display displays the entire combined frame image data on its image area; at least one synchronization unit operable to receive the merge synchronization signal from the at least one merge unit; and respective local synchronization units coupled to the graphics processors and operable to receive the merge synchronization signal and to cause the release of the frame image signal from the local frame buffer to the at least one merge unit.

2. An apparatus for processing image data to produce an image for covering an image area, comprising: a plurality of graphics processors, each graphics processor being operable to render the image data into frame image data and to store the frame image data in a respective local frame buffer; a control processor operable to provide instructions to the plurality of graphics processors; one or more merge units operable to synchronously receive the frame image data from the respective local frame buffers and to synchronously produce combined frame image data based thereon, wherein at least one of the one or more merge units is operable to produce a merge synchronization signal used by the graphics processors to release the frame image data from the respective local frame buffers to the one or more merge units, wherein the merge synchronization signal is synchronized in accordance with a display protocol defining how respective frames of the combined frame image data are to be displayed, wherein the display protocol defines at least one of a frame rate at which successive frames of the combined frame image data are displayed, and a blanking period that dictates when the combined frame image data is to be refreshed and wherein the merge synchronization signal includes transitions that are proximate to ends of the blanking periods such that the at least one merge unit initiates producing the combined frame image data for display at the end of at least one of the blanking periods; and means for transmitting the entire combined frame image data to a single display; wherein the single display displays the entire combined frame image data on its image area.

3. The apparatus of claim 2, wherein the transitions of the merge synchronization signal are substantially coincident with the ends of the blanking periods.

4. The apparatus of claim 2, wherein the merge synchronization signal includes transitions that lead the ends of the blanking periods.

5. An apparatus for processing image data to produce an image for covering an image area, comprising: a plurality of graphics processors, each graphics processor being operable to render the image data into frame image data and to store the frame image data in a respective local frame buffer; a control processor operable to provide instructions to the plurality of graphics processors; one or more merge units operable to synchronously receive the frame image data from the respective local frame buffers and to synchronously produce combined frame image data based thereon, wherein at least one of the one or more merge units is operable to produce a merge synchronization signal used by the graphics processors to release the frame image data from the respective local frame buffers to the one or more merge units; and means for transmitting the entire combined frame image data to a single display; wherein the single display displays the entire combined frame image data on its image area; the plurality of graphics processors are grouped into respective sets of graphics processors; the one or more merge units include a respective local merge unit coupled to each set of graphics processors, and a core merge unit coupled to each local merge unit; the respective local merge units are operable to synchronously receive the frame image data from the respective local frame buffers and to synchronously produce local combined frame image data based thereon; and the core merge unit is operable to synchronously receive the local combined frame image data from the

respective local merge units and to synchronously produce the combined frame image data based thereon.

6. The apparatus of claim 5, further comprising: a respective local synchronization unit coupled to each set of graphics processors and to each local merge unit; and a core synchronization unit coupled each local synchronization unit and to the core merge unit, wherein the core merge unit is operable to produce the merge synchronization signal used by the core synchronization unit and at least some of the local synchronization units to permit the respective sets of graphics processors to synchronously release the frame image data from the respective local frame buffers to the respective local merge units, and to permit the respective local merge units to synchronously release the local combined frame image data to the core merge unit.

8. A method for processing image data to produce an image for covering an image area, comprising: rendering the image data into frame image data using a plurality of graphics processors; storing the frame image data in respective local frame buffers; synchronously merging the frame image data from the respective local frame buffers to synchronously produce combined frame image data based thereon; transmitting the entire combined frame image data to a single display; displaying the entire combined frame image data on the image area of the single display; and producing a merge synchronization signal used by at least some of the plurality of graphics processors to synchronously release the frame image data from the respective local frame buffers for merging, wherein the merge synchronization signal is synchronized in accordance with a display protocol defining how respective frames of the combined frame image data are to be displayed, wherein the display protocol defines at least one of a frame rate at which successive frames of the combined frame image data are displayed, and a blanking period that dictates when the combined frame image data is to be refreshed, and wherein the merge synchronization signal includes transitions that are proximate to ends of the blanking periods such that the combined frame image data are available for display at the end of at least one of the blanking periods, and wherein the transitions of the merge synchronization signal are substantially coincident with the ends of the blanking periods.

9. The method of claim 8, wherein the transitions of the merge synchronization signal are substantially coincident with the ends of the blanking periods.

10. The method of claim 8, wherein the merge synchronization signal includes transitions that lead the ends of the blanking periods.

12. A method for processing image data to produce an image for covering an image area, comprising: rendering the image data into frame image data using a plurality of graphics processors; storing the frame image data in respective local frame buffers; synchronously merging the frame image data from the respective local frame buffers to synchronously produce combined frame image data based thereon; transmitting the entire combined frame image data to a single display; and displaying the entire combined frame image data on the image area of the single display, wherein the graphics processors can operate in one or more modes that affect at least one of (i) timing relationships between when image data are rendered, when frame image data are released from respective local frame buffers, and when frame image data are merged; and (ii) how the frame image data are merged to synchronously produce the combined frame image data, and wherein at least one of the modes is a flip animation mode providing that: (i) the local frame buffers of at least two graphics processors include frame image data that are capable of covering the image area; and (ii) the method further comprises producing the combined frame image data by sequentially releasing the respective frame image data from the at least two graphics processors.

13. The method of claim 12, wherein the flip animation mode further provides that the at least two graphics processors complete rendering the image data into the respective frame buffers prior to the ends of an integral number of blanking periods.

14. The method of claim 13, wherein the integral number of blanking periods corresponds to the number of graphics processors participating in the flip animation mode.

15. The method of claim 13, wherein the integral number of blanking periods corresponds to the number of local frame buffers participating in the flip animation mode.

Previous Doc Next Doc Go to Doc#

L6: Entry 10 of 11 File: USPT Dec 22, 1998

DOCUMENT-IDENTIFIER: US 5852449 A
TITLE: Apparatus for and method of displaying running of modeled system designs

Abstract Text (1):
A graphical, interactive debugger for a computer-based discrete-event simulation model of systems having parallel processes includes a run-time graphic user interface module in which predefined events of the executing simulation model are animated, and a run-time interface simulator module for controlling execution of the simulation model based on commands from the graphical user interface. The model and the animation are based on hierarchical directed process execution graphs in which a transaction represents one of several processes executing in parallel within the system modeled with the graphs. One transaction at a time is shown moving between nodes in one of the graphs. The animation follows the transaction as it moves between sub-models represented by each graph, switching to other graphs as needed. Once the transaction is blocked, the next transaction is displayed in context of the appropriate graph and followed until it is blocked. The screen on which animation is presented is divided into two areas. One provides the animation, the other displays the text of trace messages generated by the executing simulation model and on which the animation is based. The animation may be selectively limited to user-specified events, transactions or sub-models, providing detailed probing of the model for purposes of debugging.

Brief Summary Text (14):
In more real terms, depending on the type of system being simulated, a transaction may represent a program, a control or data signal or a data packet. A node represents manipulation of a physical resource (for example, its release or allocation), or some other processing step in a transaction's life. In a computer system model, it could represent scheduling of a microprocessor, a system bus, or a disk drive. In a model of software, it might represent a software module, a subprogram or a process control action. It could represent, in a digital electronics model, manipulation of a chip, buffer, bus, clock, register or gate.

Brief Summary Text (17):
A statistics approach is not as effective at revealing the dynamic behavior of the simulation that will enable a designer to debug the system as is animation. However, only simple simulations have so far been effectively animated. With previous methods, there has been no effective way of animating complex parallel systems. In simulations of such complex parallel systems, there are simply too many concurrent events and transactions for a designer to effectively use animation to debug a system. Moreover, there has been no effective way of animating complex parallel systems.

Brief Summary Text (20):
To animate parallel processes, the invention displays extended directed graphs representing a model of the system of parallel processes. Arcs represent flows of control and data representing the execution of a process. Nodes represent processing steps in a process. A special, sub-model node represents placement of a hierarchical subgraph within another graph. Transactions represent processes. Animation is performed for only one transaction at time by moving a transaction symbol along arcs between the nodes. Multiple transactions are not displayed simultaneously. A transaction is animated until it "blocks." Before it blocks, a transaction may flow though several directed graphs, the display switching between graphs as necessary to follow the transaction. When a transaction blocks, the graphic context switches to that of the next executing transaction.

Brief Summary Text (21):
The animation is generated from "trace" messages from an executing simulation model or, more generally, executing computer processes. In a simulating model, each executing transaction

produces trace messages upon the occurrence of predefined events. There are a number of such predefined event types each of which is animated in a distinctive way. For example, an interrupt event is animated by graphically displaying a lightning bolt, whereas a delay event is animated by turning the hands of a clock. Using distinctive animations for each event type significantly enhances the user's ability to understand what is happening in the model. The trace messages identify the nodes, arcs, graphs and event types, and determine which particular transactions are to be animated. A user may specify the animation context by selecting certain transactions or certain sub-models or graphs to be animated. This fine-grained control over animation greatly enhances debugging, as any portion of the simulation may be isolated for review.

Brief Summary Text (22):
In accordance with various other aspects of the invention, animation is displayed in a window on a video display screen coupled to a data processing system in which the simulation or other parallel processes are occurring. Another window logs flows of trace messages to allow a user to review a previously executed transaction. This log window helps a user determine the past events that lead to the model's current state.

Brief Summary Text (23):
To interact with a running model and control its animation, commands are issued by the user to stop and step the model and to change its state.

Drawing Description Text (3):
FIG. 2 is a schematic representation of components of an interactive animation and debugging process for a discrete-event simulator such as the one shown in FIG. 1 and other parallel processes.

Drawing Description Text (4):
FIG. 3 is a flow diagram of processes in a graphical runtime interface for an interactive animation and debugging program for the simulator of the type shown in FIG. 1.

Drawing Description Text (5):
FIG. 4 is a flow diagram of a back-end interactive interface for, and its interaction with, a simulator of the type shown in FIG. 1, as well as with the graphical runtime interface of FIG. 3.

Drawing Description Text (6):
FIG. 5 is a flow diagram for processes in the graphical runtime interface of FIG. 3 and the back-end runtime interface of FIG. 4 relating to producing and using trace messages as a basis for animation of a simulation carried out by a simulator or parallel process of the type shown in FIG. 1.

Drawing Description Text (10):
FIGS. 8A, 8B and 8C are illustrations for cell animations for various transactions, arcs and nodes used by the graphical runtime interface to animate an executing simulation model.

Drawing Description Text (11):
FIG. 9 is an illustration of two example sequences of animation displays which result from stepping the animation using transaction and submodel animation constraints.

Detailed Description Text (3):
Each node in the graph represents the manipulation (for example, the allocation or release) of a physical or logical resource, or some other processing step in a transaction's life. In a computer system model, a node might represent the scheduling of a microprocessor, a system bus, or a disk drive. In a software model, a node might represent a software module, a subprogram, or a process control action such as the forking of a process into subprocesses. In a digital electronics model, a node might represent the manipulation of a chip, buffer, bus, clock, register, or gate. Each arc connects two nodes and is directed from one node to the other. It represents a path along which transactions may flow from one node to another.

Detailed Description Text (14):
Referring now to FIG. 2, to provide animation of the simulator or simulation model shown in FIG. 1, a parallel process is employed. One process is the graphical run-time interface (GRI) 201, which provides a user interface to the running simulator, and a back-end runtime interface

(RTI) 203 that provides interactive access to and control over the running simulation model. The RTI is preferably part of a runtime software library that is part of a simulation language used by the simulator. This library is linked to a compiled simulation model which is written in the simulation language to produce the simulator.

Detailed Description Text (15):
The GRI and the RTI communicate via communications channels or "pipes" 250, 260 that are established when the model execution is launched. The GRI contains an animation function shown as block 205, and a debugging function at block 207. The animation function includes a graph animation function 206 which causes the dynamic update of the graphical view of the running model based upon trace messages 209 and print output 211 received from the RTI. The graph animation function 206 issues print commands to the RTI as needed to periodically update some of the animation displays.

Detailed Description Text (32):
The print or trace message at branch 329 will cause data to be displayed in the SES/scope output area at step 331. If the RTI message was a print message, the process returns to step 307 to read the next event. RTI trace messages resulting from inspector generated commands alone do not produce animation, as indicated by branch 329. Trace messages may or may not produce animation, depending on several factors, such as whether or not animation is disabled and whether or not the animation scope is constrained, as shown at 335. At step 333, the graph referenced by the event described in the trace message is identified. If the trace is animatable the relevant graph is located, step 337, and the animation is identified and displayed, step 339.

Detailed Description Text (34):
Referring now to FIG. 4, a flow diagram of the back-end interactive interface and the graphical run time interface is shown. Once the model 401 is launched at step 403, it sends a startup message to the GRI, step 405. The RTI then sends a prompt message to the GRI at step 409, and reads any commands from the GRI at step 411. Once a command is received, it is executed at step 413. The various types of commands and their execution are represented by blocks 415 to 429. Blocks 415 to 425 involve set model state commands, print commands, update flow setting commands, update trace setting commands, update break setting commands, and all other commands that are not specifically identified in the Figure. These commands cause a command output to be sent back to the GRI, as indicated by step 431. Execution of an update flow setting command, block 421, causes flow settings 433 in the execution model to be updated. Similarly, commands affecting trace settings, block 423, break settings, block 425, and step settings, block 427, result in updating trace settings 435, break settings 437, and step settings 439, respectively, in the executing simulation model 401.

Detailed Description Text (37):
Referring now to FIG. 5, a flow diagram is shown for processes in the GRI and RTI relating to trace messages being produced and used for animation of the simulator model. As the model execution proceeds, potentially large numbers of events can be generated, most with a resulting animation. The communication of these events to the GRI plus the resulting animations can slow down the effective rate of elapsed simulation time. To allow the user to raise the throughput of the simulation, communications and animations are adjusted using trace settings in the RTI and animation settings in the GRI.

Detailed Description Text (41):
The trace messages, when received by the GRI, are then taken through an animation constraint process 527 that allows constraining of animation using a general on/off facility, using a current context constraint, and using a current transaction constraint. This constraining process is analogous to a second filtering of simulation events, as illustrated by the second filter 525. Trace messages 506 flow through the constraining second filter 525 providing animated events 508.

Detailed Description Text (42):
The animation process begins at step 529 with the GRI receiving trace messages. If the user has suppressed animation, no animation occurs as indicated by blocks 531 and 533. If the user has constrained trace to the current context (submodel), at steps 535, trace for other submodels is not animated as shown by flow through steps 537 and 533. If the user has constrained trace to the current transaction, step 539, trace for other transactions is not animated as shown by flow through steps 541 and 533. As a final hurdle, an animation must be defined for the given

kind of trace, otherwise no animation is performed for the given trace event, steps 543, 545 and 533.

Detailed Description Text (43):
One exception to this GRI constraint process is when a trace message is a response to an inspector. As shown in FIG. 3 by steps 323 and 325, these trace messages are not filtered, but are passed to the inspector. This allows users to suppress "normal" trace in a desired scope, yet still be able to inspect objects within that scope without resulting in any unwanted animations.

Detailed Description Text (45):
The GRI 601 includes mechanisms available to a user for changing the state of the model, simulation, and animation. These can be considered in seven basic groups: soft buttons, indicated by block 603; inspectors, block 605; the "go until" buttons, block 607; manually entered rti commands, block 609; and the break table, block 611, the trace table, block 613, and travelling, block 615.

Detailed Description Text (46):
The soft buttons provide a means for the user to customize frequently used stepping commands by setting the soft button settings 617 using a change setting routine, indicated by block 619. These buttons accommodate four distinct configurations to control the advancing of the simulation. Each button contains a complete stepping context, including factors like animation speed, animation detail, refresh controls, and transaction and locale constraints. When changing a setting of a soft button, form 681 shown in FIG. 6B is invoked and the user can alter this stepping context information via the user interface.

Detailed Description Text (55):
As previously described in connection with FIG. 4, once the model executes up to the next step, indicated by block 642, the RTI checks it for a break or step, indicated by block 653, and for trace output and then sent to the GRI as represented by blocks 655, 657 and 661. While the trace messages are being sent, flow breaks are checked for by the RTI, as indicated by block 659, to preserve synchronization of the model to the animation. Trace messages are sent to the RTI output display window, to the update inspector function and a function for checking for animatable tracings, as represented by blocks 633, 649 and 663, respectively. Completion of a break point also causes a prompt to be sent by the RTI, as shown by block 651, to the prompt process 628 for displaying in the output window, as shown by block 633.

Detailed Description Text (56):
Animation settings 665 determine what trace is to be animated at block 663. If the trace is animatable, a model graph 667 is looked up and displayed, as represented by block 669. Then, based on the animatable trace messages, the necessary animation is looked-up and displayed, as indicated by block 671. Finally a user command 609 or use of a soft button 603 may signal the break monitor 659 to pause the simulation at the next event and enter the command loop as previously described for block 625.

Detailed Description Text (57):
Referring to FIG. 7, shown is an illustration of a user display screen created by the GRI on a monitor attached to a computer running the simulation model and the GRI. It is basically divided into two components: an animation window 701, and a text window 703. The animation window 701 provides a means of graphical communication of the current state and state transitions of an executing simulating system. The main purposes of the text window 703 are to provide a history of recent activity and to provide for user keyboard input. Users can, at any time, scroll back into this history to determine the sequence of events which resulted in the current model state.

Detailed Description Text (60):
Animation window 701 displays selected animation to reveal the states and transitions of the running model. Shown in animation window 701 of FIG. 7 is an example of a hierarchal-directed process execution graph 705 that is dynamically animating a running simulation model. Generally, these animations provide powerful, effective means of quickly communicating the current state and state transitions in an executing model. These animations take many forms which are loosely classified as transaction animation, arc traversal animation, and node animation. This corresponds to the main components of hierarchical-directed graphs--namely, transactions, arcs, and nodes. The animations are driven from events (trace, prompt, and print)

from the running model.

Detailed Description Text (61):
Referring to FIG. 8A, three examples of transaction, arc and node animation are shown. Transaction animation conveys the current transaction number and its relative location within the model. The specific animation 841 is implemented by displaying a framed "box" containing a letter "T" and the transaction number which is the unique identifier for the transaction. This box is moved along particular arcs as the transaction moves from node to node.

Detailed Description Text (62):
Arc traversal animation conveys the recent history of locale for the current transaction. The specific animation is implemented by changing the normal appearance of an arc to appearing as a "bold" arc such as shown at 843 and 844, much thicker than its normal state. As the transaction moves through the graph, it leaves a "trail" of bold arcs. This trail remains as long as the given transaction remains current.

Detailed Description Text (63):
Node animations cover the broadest category of animations. These are implemented in several forms, including blinking nodes (using color or reverse video attributes), node animations via cell animations (rapidly iterating over a set of node images), creating a graphic lightning bolt from one node to another, shown at 845, creating a moving box containing textual resource information from one point to another 847 (for animation of resources), and display text information in a boxed string 849.

Detailed Description Text (64):
The information conveyed by these animations are locale for the model event, any secondary locale for the event (lightning and resource animations), and specific changes in the state of the model, such as changes in the state of a node queue. The tables below show some of the relationships between specific animations and the model trace events which trigger them.

Detailed Description Text (67):
When a transaction is injected into a model at a source node, the user's eye is directed to the node by a series of squares that shrink rapidly around the node, providing a zooming effect that draws attention to the node of interest. This zooming effect is used at many points by the animation system to focus the user's eye on the portion of a graph in which the next animation will occur. It does not signify any particular event taking place in the model. FIGS. 8B and 8C show on each row a cell animation for a particular node. When one of these nodes is to be animated, these images are sequentially displayed to create a dynamic animation. For example, the source node appears to open when a transaction leaves it, as shown on row 825 in FIG. 8C. Similarly, when a transaction arrives at a sink node the rectangle representing the transaction shrinks and disappears into the node, as shown at row 823.

Detailed Description Text (68):
Split and fork nodes appear to open as transactions leave them, displayed in rows 827 and 811, respectively. When a parent transaction arrives at a join node it is momentarily represented as a large P that enters the node, shown on row 805. The join node appears to open as a parent transaction leaves a join node 809, after all of its child transactions have joined. If a parent transaction arrives at a join node after all of its children have joined, a small arrow travels clockwise around the join node 801. Rectangles representing child transactions shrink and disappear as they enter join nodes, displayed on row 803. When a transaction receives service at a Service node (see row 835) or is delayed at a Delay node (see row 833), the node's clock hands make one revolution to indicate activity.

Detailed Description Text (72):
When resource elements are released by a transaction, for example at a Release or Sink node, the elements are represented as a rectangle that moves from the node at which they are released to the Resource node to which they are returned. When resource elements are consumed or destroyed an "X" is added to the text of the animation to signify the consumption or destruction of the number of resource elements. As transactions move from submodel to submodel the submodel exit and entry nodes blink several times to indicate that a transaction is leaving or entering the submodel at that node.

Detailed Description Text (73):
FIG. 9 is an illustration of the differing animation sequences which result from using the

transaction and submodel animation constraints. Animation display 901 depicts a situation wherein transaction two (T2) is about to enter a subgraph called "id" 903. The animation displays 905, 907, 909 and 911 depict the next few animation displays which would appear if the current transaction animation constraint was used while stepping the model execution. The animations follow the execution of transaction two into the id subgraph 913 and through the id subgraph 905, 907 and 909 until it returns to the original calling graph 915. The animation displays 917, 919 and 921 depict the same starting situation as in 901. However, the next two animation displays result from stepping the model while using the current submodel animation constraint. The animation display first depicts the execution of transaction two 923 in 917, and then switches to the animation of transaction four 925 when continued animation of transaction two would require the display of a different subgraph.

Detailed Description Paragraph Table (1):

Animation Kinds

1. Node blinking 5. Transaction box 2. Cell animation 6. Triangle tail 3. Arc traversal 7. Boxed string 4. Lightning bolt

Associated Trace Kind Meaning Animations

create transaction creation and destruction 2 delay transaction switching out 2 discipline changes in the state of a queue fork changes in fork/join status 2,7 interrupt transaction interrupts 1,2,4,7 node node entry and exit 2 node enter node entry node return node exit 2,7 port traversal of an arc 3 resource manipulation of a resource 6,7 set effect of a set (power) statement 7 source enable trace for monitor transactions 2 submodel submodel entry and exit 1 submodel enter submodel entry 1 submodel return submodel exit 1 switch transaction posting and activation 2

CLAIMS:

1. A method of animating a system having parallel processes for the purpose of debugging the system, the system being modeled as a hierarchical collection of directed process execution graphs, said graphs representing sub-models of the system and having collections of nodes and arcs, and the parallel processes as transactions, the nodes indicating manipulation of physical or logic resources or other process steps in a transaction's life, and the arcs indicating paths along which transactions may flow from one node to another; the method comprising the steps of:

(a) executing a computer process having parallel execution threads representing parallel processes in a system;

(b) selecting one of the execution threads for animation of preselected events during the execution thread, the execution thread being represented by a transaction in a model of the system, the model including one or more hierarchial directed process execution graphs representing one or more sub-models of the system;

(c) animating the selected execution thread on a user's display screen by displaying a graph in which the transaction is located, moving a symbol on the displayed graph representing the single transaction along arcs connecting nodes to which the transaction flows, and terminating animation of the selected execution thread when the transaction is blocked by the occurrence of a predefined event;

(d) displaying user-defined inspection data on said user's display screen, wherein said inspection data comprises user-selected program operating parameters from user-selected nodes; and

(e) selecting a next transaction representing another of the parallel execution threads for animation and animating the next transaction until it blocks.

7. The simulator of claim 5 wherein the graphical user interface includes means for distinguishing a variety of predefined simulation event types and means for displaying a distinct specialized animation for each such event type.

Previous Doc Next Doc Go to Doc#